LAB 4 - Grafana

Présentation

Prometheus prend en charge deux types de règles qui peuvent être configurées puis évaluées à intervalles réguliers : les règles d'enregistrement et les règles d'alerte .

- ***recording (précalculs)*** : Les règles d'enregistrement vous permettent de précalculer des expressions fréquemment nécessaires ou coûteuses en termes de calcul et d'enregistrer leur résultat sous la forme d'un nouvel ensemble de séries chronologiques. L'interrogation du résultat précalculé sera alors souvent beaucoup plus rapide que l'exécution de l'expression d'origine à chaque fois que cela est nécessaire. Ceci est particulièrement utile pour les tableaux de bord, qui doivent interroger la même expression à plusieurs reprises à chaque actualisation.
- ***alerting (alertes)*** : Les règles d'alerte vous permettent de définir des conditions d'alerte basées sur les expressions du langage d'expression Prometheus et d'envoyer des notifications sur le déclenchement d'alertes à un service externe. Chaque fois que l'expression d'alerte génère un ou plusieurs éléments vectoriels à un moment donné, l'alerte est considérée comme active pour les ensembles d'étiquettes de ces éléments.

Gestion des alertes avec AlertManager

Pour bien comprendre : Alertmanager détecte les alertes remontées par Prometheus et décide quoi en faire. Il est possible d'avoir différents groupes, pour différents types d'alertes, pour notifier, par exemple, les trucs vraiment urgents par SMS, etc…

Les règles d'alertes se configurent côté Prometheus.

AlertManager sera ici installé sur le même serveur que Prometheus.

Installation

```
cd

wget

https://github.com/prometheus/alertmanager/releases/download/v0.18.0/alertmanager-0.18.0.linux-amd64.tar.gz

tar xvf alertmanager-0.18.0.linux-amd64.tar.gz

cd alertmanager-0.18.0.linux-amd64/

cp alertmanager /usr/local/bin/

cp amtool /usr/local/bin/
```

Puis :

```
mkdir /etc/alertmanager

mkdir /var/lib/alertmanager
```

Configuration

A – AlertManager

La configuration se passe dans le fichier */etc/alertmanager/alertmanager.yml* :

```
global:
  smtp_smarthost: 'IP.SERVEURMAIL'
  smtp_from: 'alertmanager@mondomaine.fr'
  smtp_require_tls: false #optionnel

route:
  receiver: 'alert-mails'
  group_wait: 30s         # temps d'attente avant notification du group
  group_interval: 1m      # délai par rapport aux alertes du même groupe
  repeat_interval: 5m     # attente avant répétition

receivers:
- name: 'alert-mails'
  email_configs:
  - to: $GMAIL_ACCOUNT
    from: $GMAIL_ACCOUNT
    smarthost: smtp.gmail.com:587
    auth_username: "$GMAIL_ACCOUNT"
    auth_identity: "$GMAIL_ACCOUNT"
    auth_password: "$GMAIL_AUTH_TOKEN"
```

Quand une alerte est déclenchée, elle fait partie d'un groupe.

group_wait stipule le temps d'attente pour voir si une autre alerte du même groupe se pointe éventuellement, ceci pour tout envoyer ensemble.

Si une nouvelle alerte du même groupe se présente, c'est **group_interval** qui va définir le temps entre chaque « batch ».

Et **repeat_interval** définit le temps de réémission de l'alerte si elle est toujours en cours.

On peut aussi les grouper par label avec par exemple :

`group_by: ['instance']`
qui permettra de grouper toutes les alertes d'une instance.

Il est possible de faire en fonction de match, de regex… https://github.com/prometheus/alertmanager

https://prometheus.io/docs/alerting/configuration/

Et si vous voulez bien gérer le routing, un petit outil pratique :

https://prometheus.io/webtools/alerting/routing-tree-editor/

De la même façon que pour Prometheus, on peut recharger la configuration avec :

```
curl -X POST http://localhost:9093/-/reload
```

B – Nginx

Alertmanager dispose lui aussi d'une interface Web, ce sera *https//sous.mondomaine.fr/alertmanager/*

Précision : cette interface ne remonte que les alertes en cours (celles que Prometheus lui a envoyées).

Bref, pour Nginx, classique :

```
location /alertmanager/ {
  proxy_pass http://localhost:9093/;
  proxy_http_version 1.1;
}
```

et on recharge …

Mise en place

A – en tant que commande

```
alertmanager --config.file /etc/alertmanager/alertmanager.yml --storage.path
/var/lib/alertmanager/
```

B – en tant que Service

On fait ce qu'il faut :

```
useradd --no-create-home --shell /bin/false alertmanager

chown alertmanager:alertmanager /etc/alertmanager/ -R

chown alertmanager:alertmanager /var/lib/alertmanager/ -R

chown alertmanager:alertmanager /usr/local/bin/alertmanager

chown alertmanager:alertmanager /usr/local/bin/amtool
```

Puis le fichier de service */etc/systemd/system/alertmanager.service* :

```
[Unit]
Description=AlertManager
Wants=network-online.target
After=network-online.target

[Service]
User=alertmanager
Group=alertmanager
Type=simple
ExecStart=/usr/local/bin/alertmanager \
--config.file /etc/alertmanager/alertmanager.yml \
--storage.path /var/lib/alertmanager/ \
--web.external-url=https://sous.mondomaine.fr/alertmanager/ \
--web.route-prefix="/"

[Install]
WantedBy=multi-user.target
```

La aussi, j'ai ajouté les deux flags *web.external-url* et *web.route-prefix*.

Et pour finir :

```
systemctl daemon-reload

systemctl enable alertmanager.service

systemctl start alertmanager
```

Puis :

```
service alertmanager status
```

C – Ménage

```
cd

rm alertmanager-0.18.0.linux-amd64 -r

rm alertmanager-0.18.0.linux-amd64.tar.gz
```

Utilisation

La première chose à faire est déjà de s'assurer que l'envoi des alertes fonctionne correctement. Une fois de plus, on va « Curler » :

```
curl -H "Content-Type: application/json" -d '[{"labels":{"alertname":"Test"}}]' localhost:9093/api/v1/alerts
```

C'est d'ailleurs de cette façon que Prometheus déclenchera Alertmanager. Et peu de temps après, un mail doit arriver.

Liaison avec Prometheus

On va connecter Prometheus et Alertmanager dans */etc/prometheus/prometheus.yml*, où l'on va modifier la partie alerting et la partie rules:

```
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - localhost:9093
    scheme: http
    timeout: 10s

rule_files:
- 'rules/*'
```

Et on recharge Prometheus (si l'API du prometheus et exposée) :

```
curl -X POST http://localhost:9090/-/reload
```

Règles

On va d'abord créer un répertoire pour les accueillir :

```
mkdir /etc/prometheus/rules
```

Puis dans ce répertoire, un fichier *alerts.yml* ou n'importe quel autre nom… :

```
groups:
- name: toto
  rules:
  - alert: InstanceDown
    expr: up == 0
    for: 1m
    labels:
      severity: critical
    annotations:
      summary: "L'instance {{ $labels.instance }} est down"
      description: "Le Job: {{ $labels.job }} signale que {{ $labels.instance }}
est down depuis deja 1 minute. Tu attends quoi ?"
  - alert: DiskFull
    expr: node_filesystem_free_bytes{mountpoint="/",instance="localhost:9100"} /
1024 / 1024 / 1024 < 20
    for: 1m
    labels:
      severity: warning
    annotations:
      summary: "Et zut, on arrive a moins de 20Go sur {{ $labels.instance }}"
      description: "Il reste precisement {{ $value }}"
```

Dans ce fichier, deux alertes simples pour l'exemple. On peut en mettre autant qu'on veut.

Pour le groupe, rien de sorcier, on lui donne un nom et ensuite les alertes associées. Pour les alertes, chaque fois la même structure :

- Le nom de l'alerte, sans espace.
- L'expression sur laquelle filtrer. Pour la première, c'est simple. Pour la seconde, je filtre sur un serveur en particulier et l'alerte se déclenche si le résultat est inférieur à 20.
- *For* indique pendant combien de temps la condition doit être remplies pour que l'alerte soit déclenchées. Elle est Pending pendant ce laps de temps.
- Ensuite, on peut rajouter des labels.
- Puis le contenu ou il est possible de reprendre les variables.

On peut vérifier le fichier avec :

```
promtool check rules /etc/prometheus/rules/alerts.yml
```

Puis on recharge Prometheus :

```
curl -X POST http://localhost:9090/-/reload
```

Envie d'idée pour de nouvelles règles ?

Allez voir par ici : https://awesome-prometheus-alerts.grep.to/rules.html

Tests

Il n'y a plus qu'à tester et rien de plus simple !

Coupons Node Exporter sur une machine ou il est installé.

```
systemctl stop node_exporter
```

On attend quelques instants et dans l'interface de Prometheus *http//sous.mondomaine.fr/prometheus/*, dans la section *Alerts*, on doit voir l'alerte passer en orange (et le détail indique *Pending*)

Puis, une fois le délai *for* dépassé, elle passe en rouge (*Firing*) et est transmise à Alertmanager. Elle sera visible dans l'interface *https//sous.mondomaine.fr/alertmanager/* et on reçoit le mail également.

Exercice

1- Ajoutez une alerte pour envoyer un email si la charge moyenne du système d'une minute dépasse le 60 % toutes les 20 secondes. Indice : pour stresser le système exécutez la commande suivante : **`while true ; do df ; done`**

Solution

```
  - alert: Trop_2_load
    expr: node_load1 >= 0.6
    for: 20s
    labels:
      severity: critical
    annotations:
      summary: "{{ $labels.instance }} - trop de load"
      description: "Le server en prend plein de charge "
```

2- Ajoutez une alerte pour envoyer un email si Prometheus a redémarré plus de deux fois au cours des 15 dernières minutes. C'est peut-être un crashloop.

Solution

```
  - alert: PrometheusTooManyRestarts
    expr:
changes(process_start_time_seconds{job=~"prometheus|pushgateway|alertmanager"}[15m
]) > 2
    for: 0m
    labels:
      severity: warning
    annotations:
      summary: Prometheus too many restarts (instance {{ $labels.instance }})
```

```
      description: " Prometheus a redémarré plus de deux fois au cours des 15
dernières minutes. C'est peut-être un crashloop.\n  VALUE = {{ $value }}\n  LABELS
= {{ $labels }}"
```

3- Ajoutez une alerte pour vérifier si l'Hôte à court de mémoire (< 20 % restants)

Solution

```
  - alert: HostOutOfMemory
    expr: node_memory_MemAvailable_bytes / node_memory_MemTotal_bytes * 100 < 10
    for: 2m
    labels:
      severity: warning
    annotations:
      summary: Host out of memory (instance {{ $labels.instance }})
      description: "Node memory is filling up (< 20% left)\n  VALUE = {{ $value
}}\n  LABELS = {{ $labels }}"
```

4- Ajoutez une alerte pour vérifier si une hôte a un charge CPU élevé dépassant le 8%

Solution

```
  - alert: HostHighCpuLoad
    expr: 100 - (avg by(instance) (rate(node_cpu_seconds_total{mode="idle"}[2m]))
* 100) > 80
    for: 0m
    labels:
      severity: warning
    annotations:
      summary: Host high CPU load (instance {{ $labels.instance }})
      description: "CPU load is > 80%\n  VALUE = {{ $value }}\n  LABELS = {{
$labels }}"
```